

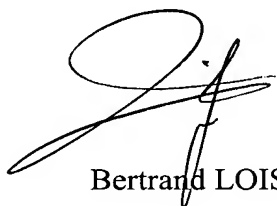
10/539456

JC20 Rec'd PCT/PTO 17 JUN 2005

CERTIFICATION OF TRANSLATION

I, Bertrand LOISEL, of CABINET PLASSERAUD, 65/67, rue de la Victoire, 75440 PARIS CEDEX 09, FRANCE, do hereby declare that I am well acquainted with the French and English languages, and verify that the document attached is a true English language translation of the text of International Patent Application no. PCT/FR2003/003225.

Dated this 21th day of April 2005.



Bertrand LOISEL

JC20 Rec'd PCT/PTO 17 JUN 2005

CONFIDENCE COMMUNICATION METHOD BETWEEN TWO UNITS

The present invention relates to personal computer terminals which enable users to access online services.

5 Terminals of this type may, in particular, be
telephones using the wireless application protocol
(WAP), office computers, portable computers or personal
digital assistants (PDA), sharing the common
characteristic of being connected to a digital data
10 network which, in many practical cases, is a network
operating according to the IP protocol ("Internet
protocol"), in particular the Internet.

Various applications can be installed in these
terminals. Among these applications, a distinction is
15 frequently made according to various criteria such as
their origin, the degree of confidence assigned to them
by an administrator, etc., resulting in different
capacities for certain applications in relation to
others.

20 For example, in systems operating under the
"Unix" operating system, the execution rights of
applications in the "setuid root" class are the maximum
rights, at administrator level, whereas the execution
rights of other applications are simply the rights of
25 the user launching the application. In another example
involving Web browsers which comprise a virtual Java
machine, the applications, referred to as "applets",
downloaded from a given website, are limited in terms
of their capacities to access the network, i.e. they
30 can only transmit HTTP ("Hypertext transfer protocol")
requests to this website.

Some of these execution rights of the
applications are purely local. This is the case, for
example, with the right to take control of the screen

of a terminal, and the right to know all the keys pressed on the terminal keyboard, even for other applications.

However, other execution rights can be observed
5 remotely. This is the case, for example, with the right to transmit any IP packets, including IP packets which do not comply with the most common transport protocol formats, i.e. TCP ("transmission control protocol") or UDP ("user datagram protocol"). In Unix systems, this
10 right is not granted to applications which are not in the "setuid root" class. By using this difference in the capacity to transmit requests, a remote observer such as a server can determine that a given packet has been transmitted by an application in the "setuid root"
15 class: if it observes that this packet does not comply with the TCP or UDP format, it must necessarily be an application in the "setuid root" class; if not, it may involve an application without special rights.

In the case of applets in browsers on personal
20 computers, the capacities to send HTTP requests are limited to the single site from which the applet was downloaded. For each received HTTP request, a Web server may therefore infer that it originates either from an applet present on the site or from a different
25 application (for example the browser). However, in any case, the requests received by a Web server do not originate from "foreign" applets present on other sites.

Interest is focused here on the problem of
30 knowing how a server may, in a secure manner, capture the consent of the user with regard to a given question. The question to be posed to the user must be presented to the user via an application on the user's terminal. The application captures the consent (or
35 refusal) of the user, then transmits a corresponding indication to the server.

The server therefore receives messages on the network and interprets them as the consent (or refusal) of the user. To do this, it must assume that the application has in fact presented the question to the user and has captured the user's consent in all honesty. The server therefore assumes that the application is not a "Trojan horse" which has, for example, presented a different question to the user, or has simply not presented the question to the user, but simulates the user's consent. To protect the user against any programs of the "Trojan horse" type, this assumption of confidence must be assured.

Several means exist for reasonably satisfying this assumption of confidence in the application.

Some applications are recognized as "confidence" applications. An application of this type is, for example, the WAP browser. A server may have confidence in a WAP browser for it to display a page posing a question to the user and wait for the user to enter his response.

In the case of a "closed" terminal (e.g. a Minitel), the applications presented on the terminal are known and cannot be changed during the lifetime of the terminal. All these applications are recognized as "confidence" applications.

The openness of a terminal refers to the facility offered to the user to install, and often download, new applications which are intended to be run by the terminal itself. Examples of "open" terminals which integrate this facility are:

- * application-downloading telephones, for example of the Java MIDP type ("Mobile Information Device Profile", Sun Microsystems, Inc.);
- * browsers with scripting functionalities, for example of the WMLScript type (see "WAP WMLScript

Language Specification", version 1.1, WAP Forum, November 2001) or ECMAScript (also referred to as JavaScript, see "ECMAScript Language Specification", Standard ECMA-262, third edition, December 1999), or browsers which accept applets;

* most of the PDAs operating under operating systems such as PalmOS, WindowsCE, Symbian, etc.;

* office or portable computers.

"Semi-open" terminals are open terminals in which certain functionalities are not directly accessible to the applications installed by the user or downloaded. For example, in a terminal whose only "openness" is ECMAScript, downloaded applications cannot access all the functionalities of the network (for example, transmission of any given IP packets). These functionalities may be accessible in an indirect and controlled manner. For example, an ECMAScript function may order the loading of a page via HTTP, which entails using the network, but in a controlled manner.

In "semi-open" terminals, there is coexistence:

* of applications regarded as "confidence" applications, for example because they have been factory-installed by the terminal manufacturer, or because of the guarantee obtained by means such as the electronic signature of the application, etc.

* and of other applications which may be installed on the terminal by the user himself, at his own discretion, but which do not access the same rights as the confidence applications.

On the other hand, "fully open" terminals are open terminals in which all functionalities are accessible to the downloaded applications. The notion of openness of a terminal depends largely on the context in which it occurs. For example, different

layers of the OSI model (link/network/session/transport/etc.) may have different degrees of openness.

Interest is focused here on the functionalities which can be observed remotely from a server, i.e. network functionalities. In this context, the "semi-open" character of a terminal generally implies that execution rights which can be observed remotely and which are accessible to confidence applications are not accessible to non-confidence applications (for example the right to transmit requests other than HTTP on an IP network). This enables a server to distinguish, from among the requests received by it, those which originate from confidence applications and those which originate from other applications.

"Applets" which the user installs at his own discretion do not necessarily afford the confidence to access any given server. However, the restriction of the requests of each applet to the site from which it was downloaded enables a website to maintain control over the applets which can transmit requests to it. It is therefore reasonable for the server to assume that the applications presenting questions to the user are not Trojan horses. These applications are therefore "confidence" applications, but for one website only.

In open terminals, the possibility that a program may behave in a deceptive manner vis-à-vis the user (Trojan horse) must be taken into account. Thus, nothing can guarantee to a server that a request actually originates from the user and not from a program which has simulated the user's consent in the network. This risk undermines the confidence which the server may have in the data which it receives from a client. The assumption that the requests addressed to the server reflect the actions of the user is not reasonable if a Trojan horse has the facility to send them instead of the user.

The conventional response to the Trojan horse risk is to limit the capacities of the non-confidence applications.

The limitation of the transmission of frames from semi-open terminals is normally imposed in an extremely strict manner. Only confidence applications are authorized to transmit certain frames. This distinction is used so that the server does not accept, as representative of the user's consent, frames transmitted by non-confidence applications which may betray the user.

It therefore becomes impossible for a non-confidence application to transmit frames to a server. It is, in particular, impossible for this application to prove the user's consent to this server. For example, it is impossible for a non-confidence application to propose to the user to make a payment using an electronic commerce server.

For an "applet", which is restricted to being able to transmit requests only to the website from which it was downloaded, confidence is conferred upon this server only. It is therefore possible for this applet to capture the user's consent and to transmit the result to the website from which it was downloaded. It is therefore - reasonably - assumed that the server has never proposed to download "Trojan horse" type applications.

Cryptography-based systems exist to generate electronic signatures. An example is described in the specification "WAP WMLScript Crypto Library", WAP Forum, June 2001. These systems can be used to capture the user's consent, wherein they assume that the system is semi-open, i.e. in this case that the functions for accessing cryptographic keys are not directly available to non-confidence applications. Access to cryptographic keys is managed by a specific software component, which

will be referred to as the "electronic signature component", and which is responsible for capturing the user's consent on behalf of the application. This component itself performs the following concatenation
5 of operations on behalf of the non-confidence applications:

- * display the text to be signed on-screen;
- * wait for confirmation from the user
- * if confirmation is received, use the
10 cryptographic keys of the user to sign the displayed text;
- * if not, do not sign the displayed text.

This therefore allows non-confidence applications to obtain an electronic signature of the user's consent
15 via the electronic signature component. This method allows the server to obtain the user's consent concerning any given text.

It must also be assumed that the terminal is not fully open. If it were possible for a non-confidence
20 application to gain direct access to cryptographic functions, it would not be possible to know whether the call for cryptographic functions was actually preceded by a display of the entire text to be signed or whether the terminal actually waited for the user's consent
25 before proceeding with the signature.

Furthermore, this method implements cryptographic techniques which may prove costly in terms of execution time, the size of messages exchanged on the network and power consumption (an important consideration in the
30 case of portable terminals). Moreover, legislation governing cryptographic techniques may possibly restrict the facility to make use of this method.

It is therefore desirable to provide a behavior which is more or less equivalent in terms of openness

to non-confidence applications, but without using cryptography.

An object of the present invention is to enable a "non-confidence" application in a semi-open environment
5 to capture the user's consent to a given question, and to report this to a remote server, proving to it that this has been done in an honest manner.

The invention therefore proposes a method for communication between a first unit and a second unit
10 via a telecommunications network, wherein the first unit comprises a first family of applications and a second family of applications which has capacities to communicate on the network extending beyond the communication capacities of the applications of the
15 first family. According to the invention, the method comprises the following steps:

- 20 /a/ a confidence component belonging to the second family of applications obtains the statement of a question to be posed to a user of the first unit in the context of the execution of an application of the first family;
- /b/ the confidence component presents the question via a user interface and captures a response from the user; and
- 25 /c/ for at least one type of response from the user, the confidence component transmits to the second unit, via the network, at least one message identifying the question presented and indicating the response captured, said message being
30 transmitted under conditions which are inaccessible to the applications of the first family.

Another aspect of the present invention relates to a confidence software component for implementing the
35 above method in said first unit, and also a

communications terminal incorporating such a confidence software component. This confidence component belongs to the aforementioned second family of applications and includes instructions to control the following steps
5 during its execution in the first unit:

- /a/ obtain the statement of a question to be posed to a user of the first unit in the context of the execution of an application of the first family;
- 10 /b/ present the question via a user interface and capture a response from the user; and
- /c/ for at least one type of response from the user, transmit to the second unit, via the network, at least one message identifying the question presented and indicating the response captured,
15 said message being transmitted under conditions which are inaccessible to the applications of the first family.

Other features and advantages of the present invention will be shown in the description below of
20 non-limiting embodiments, with reference to the attached drawings, in which figures 1 and 2 are diagrams of a system implementing the invention.

It is desired to enable a remote unit such as a server 1 to obtain, in a secure and flexible manner,
25 the consent of the user of a semi-open terminal 2 relating to a given question. The consent may be obtained by confidence applications 3, as in the case of browsing, but also from non-confidence applications 4, which have more restricted (or non-existent)
30 capacities to communicate on the telecommunications network R used to dialogue with the server 1.

The present context is that of a terminal 2 making a distinction between confidence applications 3 and non-confidence applications 4. This distinction
35 entails separate capacities for the transmission of

frames or requests on the network R. The non-confidence applications 3 are limited in terms of the frames that they can transmit which, in the diagram shown in figure 1, is represented by a control layer 5 which
5 forms part of the network access resources 6 with which the terminal 2 is equipped.

The control layer 5 checks that certain characteristics are met by the frames transmitted by the non-confidence applications 4. If these
10 characteristics are met, the control layer allows the frames to pass. If not, it can either not allow them to pass to the network R and report this to the non-confidence application 4 which transmitted them, or
15 modify the frames so that they comply with the constraints of the non-confidence applications. In this last case, the frame then loses its credibility in the eyes of the server 1, which will not handle it.

The invention uses this control layer 5 (the presence of which may only be implicit and may result
20 from characteristics of the operating system or, more generally, the applications-running environment in the semi-open terminal) to prevent a non-confidence application 4 from itself transmitting requests which would provide a server with proof of the consent of the
25 user relating to the question asked. Such an application cannot therefore itself capture the user's consent in a form which can be handled by the server 1.

A confidence software component 8, the behavior of which has previously been assured as "honest", is
30 therefore introduced in the terminal 2 between the non-confidence application, the server and the user. In practice, this assurance often originates from the manufacturer of the semi-open terminal. The confidence component 8 cannot be replaced or modified by a non-
35 confidence application, this being assured by the semi-open system itself, for which a confidence application

must remain a confidence application. There is therefore no risk that the component 8 will behave like a Trojan horse. A main role of the confidence component 8 is to capture the user's consent on behalf of one or more non-confidence applications 4 by means of a user interface 9 of the terminal.

The confidence component 8 is not limited in the requests that it can transmit, or it is at least subject to less severe restrictions than the non-confidence applications 4. In the example shown schematically in figure 2 above, it is not controlled by the control layer 5.

Interest is focused on an application 3 or 4 that wishes to prove to a remote server 1 that it has obtained the user's consent for a given question. It initially has the question statement and addressing data which enable it to contact the remote server, for example an indication of the URL ("Uniform Resource Locator") type.

The communications of these applications 3, 4 are subjected to the following rules:

- the applications can carry out remote communications via the resources 6 and the network R, but these communications are limited by the semi-open operating system which incorporates a logical control layer 5;
- any remote server 1, knowing these applied limits, can determine whether the messages that it receives originate from confidence applications or not by examining whether the limits are applied;
- the confidence component 8 has the facility to carry out communications outside the limits imposed on the non-confidence applications 4, but also within these limits if it wishes. In this respect,

it can be regarded as belonging to the same family as the confidence applications 3.

A non-confidence application that wishes to obtain the user's consent to a given question and to
5 prove this consent to a remote server 1 provides the confidence component 8 with the question statement and the address of the server. The confidence component 8 then presents the question to the user by means of the interface 9. The user's decision (to accept or refuse,
10 the absence of a response after a certain period of time being able to be interpreted as a refusal) is captured by the confidence component.

If the decision captured is a consent, a request outside the limits applied to the non-confidence
15 applications is sent by the confidence component to the server at the address previously indicated by the application 4. This request contains:

- the question statement,
- the user's response.

20 The server 1 implicitly or explicitly checks that the request has actually been transmitted outside the limits applied to the non-confidence applications, and responds to this request following validation. The response to the request is finally transmitted by the
25 confidence component 8 to the non-confidence application 4.

In the case of refusal on the part of the user observed by the confidence component 8, the latter can transmit a response indicating the refusal directly to
30 the application 4. The negative response of the user is only optionally transmitted to the server 1 in this case.

If it has confidence in the "confidence component" 8, the remote server 1 is assured that the

requests outside the limits that it receives in fact correspond to questions which have been posed to the user, and that the user's choice has been correctly captured. A non-confidence application cannot simulate
5 this behavior. The Trojan horse risk has therefore been eliminated.

If the check by the server 1 of the request deemed to indicate the user's consent shows that it has been transmitted within the limits applied to non-
10 confidence applications, this request is not interpreted as being representative of the user's consent. This refusal of the server can optionally be reported back to the terminal.

Naturally, the question posed to the user may
15 invoke a response of any type, more elaborate than "yes/no". The question may, in particular, be presented as a form in which a plurality of entries are to be supplied by the user. In this case, the different entries provided by the user can be transmitted to the
20 server after the confidence component 8 has requested and obtained a validation from the user.

In the preceding description, the non-confidence application 4 itself generates the question text. If it is preferred that the server 1 generates the question
25 text, it is possible, for example, to proceed as follows:

- an non-confidence application 4 submits to the confidence component 8 the address of a server 1 (for example a URL) and an appropriate request to
30 send to it to obtain the statement of the question to be posed;
- the confidence component 8 transmits the request via the network R in order to request the question statement from this server 1. The request is
35 preferably sent via the control layer 5 in order to

guarantee that it is within the limits authorized for non-confidence applications 4;

5 - the server 1 forwards the question statement, in association with a reference to be re-used later during the transmission of the user's consent;

 - the confidence component 8 presents the question to the user as previously;

 - the user makes his decision;

10 - the user's decision is captured by the confidence component 8;

 - in the case of consent, the confidence component 8 transmits a request to the server 1, this time outside the limits imposed on non-confidence applications, and including the statement reference and stipulating that the user has indeed given his consent (the reference can be optional, in which case the confidence component repeats the question statement in the request transmitted at this stage; generally, it will suffice that the question posed is adequately identified in the message transmitted to the server to indicate the user's consent);

20 - the server 1 validates the request by checking that it has actually been received outside the limits imposed on non-confidence applications, and responds to this request;

25 - the response is transferred to the application which initiated the request.

30 As it has been assured that the request originating directly from the non-confidence application 4 has been passed via the control layer 5, the server 1 rests assured that the requests outside the limits which it receives from the confidence component 8 actually result from an explicit consent of the user.

In a specific embodiment of the invention, the terminal has a virtual Java machine which may correspond to the module 6 shown in figures 1 and 2. The virtual machine enables the execution of downloaded applications written in the Java programming language developed by Sun Microsystems, Inc. All the Java language instructions are executed by the virtual machine, using system functions following a certain control. Java applications involve a semi-open environment, since there is no uncontrolled invocation of system functions.

The non-confidence application 4 is then written in Java language.

In this embodiment, the protocols used for the exchanges of the terminal 2 on the network R are HTTP (RFC 1945 ("Request For Comments"), published in May 1996 by the IETF ("Internet Engineering Task Force")), TCP (RFC 793, IETF, September 1981) and IP (RFC 791, IETF, September 1981). The limit applied to non-confidence applications is that they cannot address requests to URLs of the following type: "http://<server>/<path>/consent?<continuation>", where <server> is any given server name, <path> is a series of character strings in the form "directory_1/directory_2/.../directory_n" and <continuation> is any string of characters. This limit is of course an example, and any other limits can be imposed. The service is hosted by an HTTP server.

The confidence component 8 can then be implemented in the virtual Java machine by the *UserConfirmation* class. It is accessible from Java applications 4 by a class function: *InputStream UserConfirmation.ask(String url, String question)*, the function of which is as follows. When a non-confidence application 4 invokes the function *UserConfirmation.ask(String url, String question)*:

- the confidence component 8 opens a window or actually takes control of the terminal on the application currently running;
- the question whose statement is given by the string of characters "question" is displayed on-screen, and two choices are offered to the user, i.e. "OK" and "Cancel";
- if the user gives his consent, by selecting "OK":
 - * the confidence component 8 sends on the network R the HTTP request formed by the concatenation (i) of the URL passed as a parameter ("url"), (ii) of the string "/consent?question=", (iii) of the statement of the question posed to the user (encoded in an encoding format in the URL x-www-urlencoded), and of the string "&responseOK". This behavior is of course only an example which corresponds to the limitation imposed on the requests originating from Java applications. A server is assured by this combination that the requests sent at this stage by the confidence component could not have been sent by the Java applications, thereby meeting the requirement;
 - * if the confidence component 8 then receives the response from the server 1 (or an exception if the server is not available), it returns an *InputStream* object to the invoking application 4 which enables this application to know the response of the server;
- if the user does not give his consent, by selecting "Cancel":
 - the confidence component 8 forwards an exception to the invoking application 4.

This particular embodiment can be illustrated by the case in which the server manages a micropayment service which makes payments online on behalf of the user simply with the consent of said user. The payments
5 are debited to an account corresponding to the user. If it receives a payment order, this service therefore
~~wants to make sure that this order is actually~~
confirmed by the user, and does not originate from a malicious Java program which has not presented any
10 question to the user, or indeed which has presented the user with a misleading question. This service is of course an example, and any other service requesting the user's consent can be implemented thanks to this technique (document publication, file management,
15 messaging, etc.).

In this example, the payment service controls the "payment.com" website. If a non-confidence application wishes to offer a payment to the user, it invokes the UserConfirmation.ask function, providing it with the
20 following parameters:

- * as the URL: `http://payment.com/payment`,
- * as the question statement: "Pay €1 to Acme Co.?"

The confidence component 8 takes control of the terminal 2, and asks the user "Pay €1 to Acme Co.?
25 OK/Cancel". If the user selects the "OK" link, the confidence component transmits the request
"`http://payment.com/payment/consent?question=pay+€1+to+Acme+Co.?&response=OK`" and transmits the response from the server to the invoking application 4 with a return
30 handshake.

If the user selects the "Cancel" link, the confidence component 8 does not transmit any request and returns an exception to the invoking application 4.

If an application 4 attempts to request directly
35 the page "`http://payment.com/payment/consent?question`

=pay+€1+to+Acme+Co.?&response=OK", this request is refused by the limitation imposed on non-confidence applications.

5 The method as claimed in the invention can be further illustrated by the case in which the server manages an electronic commerce service. In the context of a service of this type, the customer is prompted to complete an order form. This form is to be sent according to the HTTP POST method to the address
10 "http://service.com/order".

The confidence component can then be implemented in the virtual Java machine. It is accessible from Java applications via the following function:
"UserConfirmation.askForm(String url, byte[] form)".

15 If this function is invoked by a Java application 4, the confidence component 8:

- 20 * displays on-screen the form contained in the "form" table supplied as a parameter of the function. This form is, for example, in XML format;
- * allows the user to complete the form fields and asks the user to validate this by selecting "OK" or "Cancel" at the end of the form;
- 25 * sends an HTTP POST request, if the user validates the form, to the URL "url+/consent?", this request containing the form which was presented to the user, along with the data entered by the user in the different fields.

30 If a non-confidence Java application 4 attempts to access directly the address "url+/consent?", the request will be refused by the control layer.

Moreover, an application could attempt to mislead the user by making him complete a form containing the same entries as the authentic form, but with different

headings. This attack is equally thwarted by the fact that the form is transmitted to the server 1 by the confidence component 8. In this way, the server 1 can actually check that the form completed by the user is indeed a legitimate form.

To clarify the description, a simple example has been taken of the limitation imposed on non-confidence applications, i.e. certain URLs are not accessible, this being controlled at the time of transmission of a request. Nevertheless, any other limitation would be acceptable.

In particular, complete blockage of any access to the network R for non-confidence applications 4 could be used, a selective blockage authorizing only requests to the originating website of a downloaded application, etc.

The limitation may also relate to a specific marking associated with either non-confidence applications 4 or confidence applications 3. Each request originating from a non-confidence application 4 which is transmitted on the network R and is destined for the server 1 is then forced by the control layer 5:

- /1/ either to include a marking associated with the family of non-confidence applications,
- /2/ or not to include a marking associated with the family of confidence applications, this marking then being included in at least some of the requests transmitted on the network R and originating from confidence applications.

In case /1/, the confidence component 8 does not place the marking in the requests transmitted to indicate the user's consent, thereby assuring the server 1 that this consent actually originates from the user. The confidence component 8 may, on the other hand, mark the request transmitted on the network R to

obtain the statement of the question to be posed in the event that this statement is not supplied directly by the application 4.

Conversely, in case /2/, the confidence component 5 8 places the marking in the requests transmitted to indicate the user's consent, and, if appropriate, it does not mark the requests transmitted on the network R to obtain the statement of the question to be posed.

In the example in which the confidence component 10 8 forms part of a virtual Java machine 6, the marking of case /1/ consists, for example, in that the "User-Agent" header field of the HTTP requests (cf. section 10.15 of the aforementioned RFC 1945) contains a specific string such as "Non-confidence application: VM 15 Java 1.2" which indicates by its presence that the request does not originate from a confidence application. An opposite statement can be provided in case /2/.